BAB III
METODOLOGI PENELITIAN

3.1. Teknik Pengumpulan Data

Metode yang digunakan dalam proses pengumpulan data dan penelitain ini adalah sebagai berikut :

3.1.1. Pengamatan *(Observasi)*

Pengumpulan data dengan cara mengamati dan mencatat secara langsung kegiatan yang terjadi pada saat proses parkir.

3.1.2. Wawancara *(Interview*)

Metode ini dilakukan dengan cara melakukan tanya jawab secara langsung kepada pihak yang terkait terhadap permasalahan yang berhubungan secara langsung. Pada tahap ini peneliti melakukan wawancara kepada seorang guru yang ditunjuk untuk menjadi narasumber di SMPN 10 Bandar Lampung.

3.1.3. Tinjauan Pustaka

Penyusun melakukan tinjauan pustaka yaitu dengan mengumpulkan data dari buku-buku referensi, dan sumber-sumber lain yang dapat mendukung dalam pembuatan penelitian ini. Dalam penelitian ini peneliti mencari referensi dari buku dan jurnal-jurnal yang terkait dengan judul Rekayasa Perangkat Lunak, Analisis dan Desain Informasi, Pendekatan Android, *Black Box,* dan jurnal mengenai sistem parkir.

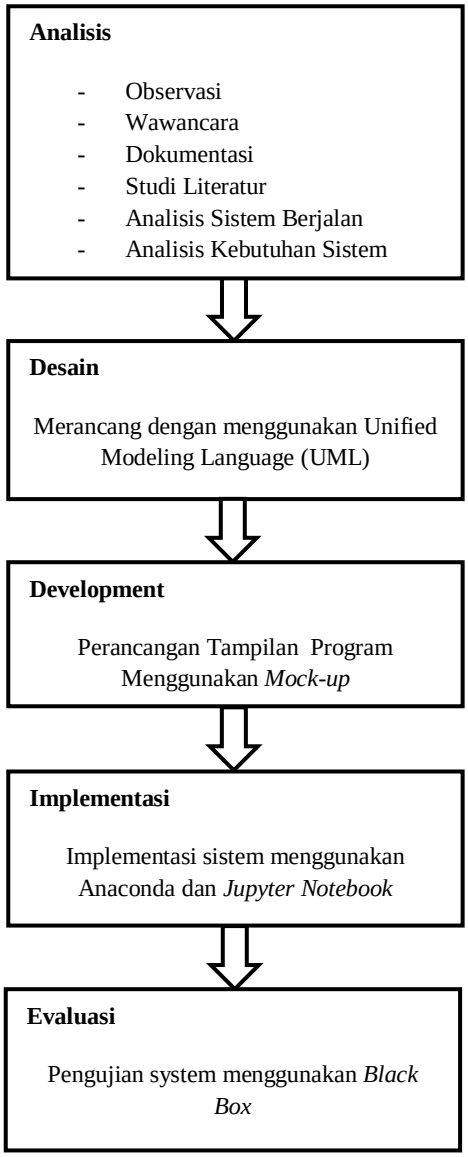
3.1.4. Dokumentasi (*Documentation)*

Dokumentasi dapat diartikan sebagai sesuatu yang tertulis, tercetak atau terekam yang dapat dipakai sebagai bukti atau keterangan. Dokumentasi dilakukan untuk mengumpulkan data yang bersumber dari arsip dan dokumen yang ada hubungannya dengan masalah yang dibahas.

3.2. Tahapan Pengembangan Sistem

Dalam pengembangan system menggunakan *UCD* adalah metodologi pengembangan perangkat lunak yang ditujukan untuk meningkatkan kualitas

perangkat lunak dan tanggap terhadap perubahan. Berikut gambar tahapan *UCD* yang diajukan penulis dapat dilihat pada gambar 3.1 dibawah ini :



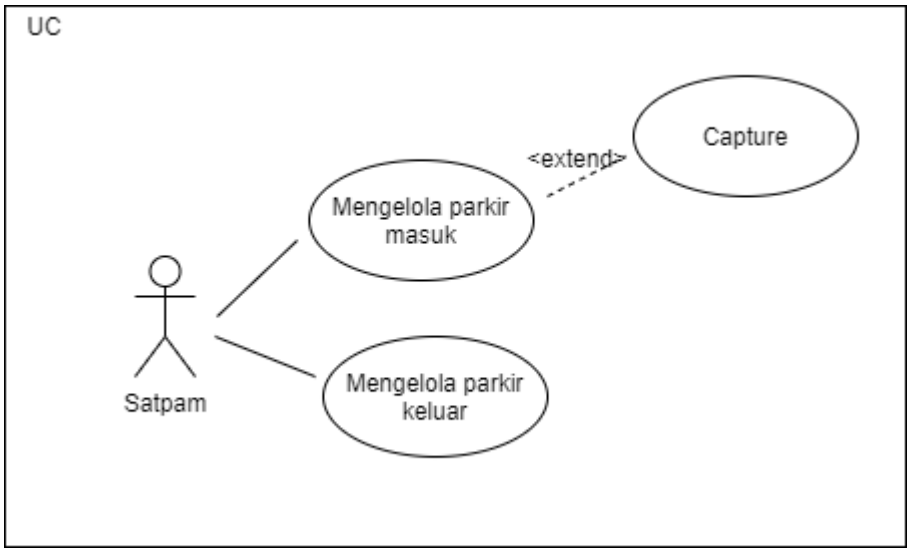
Gambar 3.1 Tahapan Penelitian

3.3. Gambaran Sistem

Penggambaran yang dipilih dalam penelitian ini adalah UML yaitu :

3.3.1. Usecase Diagram

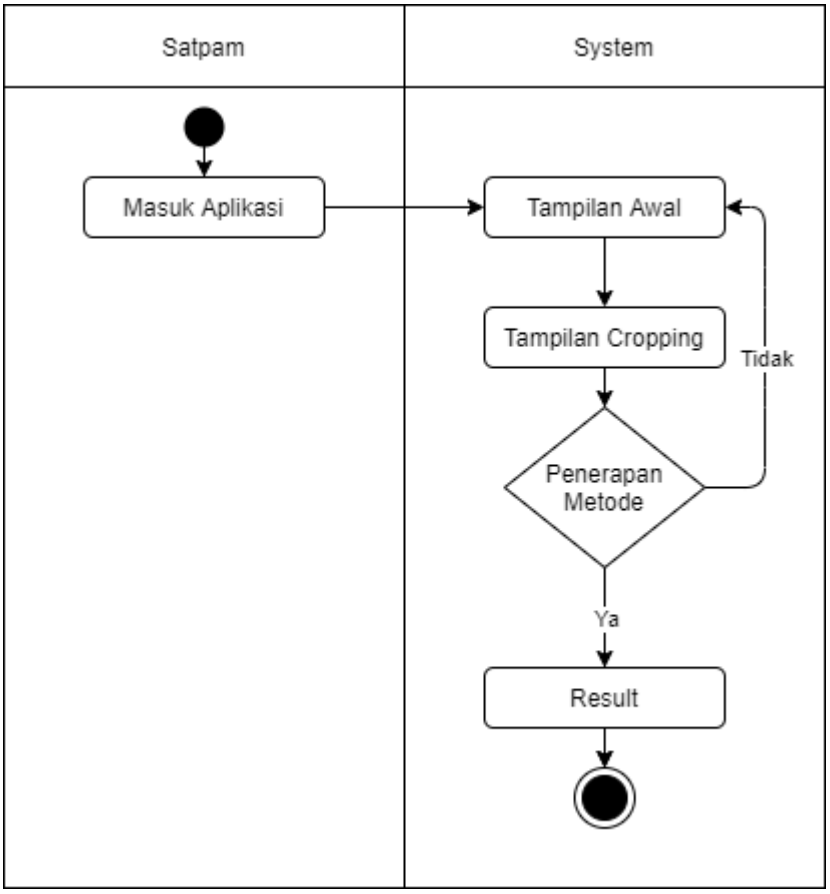
Use case Diagram mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Adapun gambar *Usecase* diagram dapat dilihat pada gambar 3.2:



Gambar 3.2 *Usecase Diagram*

3.3.2. Activity Diagram

Activity diagram atau diagram aktivitas menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses. Berikut ini adalah activity diagram sesuai dengan alur dalam aplikasi yang dibangun. Adapun gambar *activity* diagram dapat dilihat pada gambar 3.3:



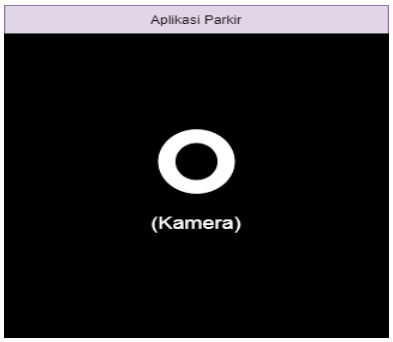
Gambar 3.3 *Activity* Diagram

3.4. Desain Program

Desain merupakan penggambaran system yang akan dibangun, berikut adalah rancangan tampilan system :

3.4.1. Rancangan Tampilan Awal

Tampilan atau halaman awal digunakan untuk open kamera dan juga capture gambar dengan menekan tombol (space), maka akan otomatis tersimpan gambar yang sudah diambil.



Gambar 3.4 Rancangan Tampilan Awal

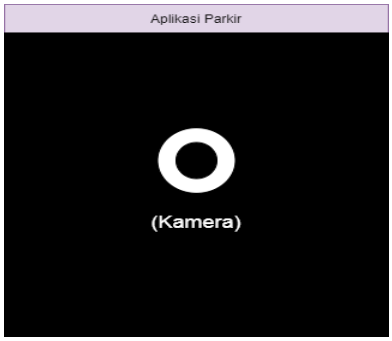
3.4.2. Rancangan Tampilan Cropping

Tampilan Cropping digunakan untuk crop gambar otomatis sesuai dengan ukuran yang sudah ditentukan. Kemudian akan menampilkan gambar yang asli (kanan) dan gambar yang sudah di crop (kiri).

Gambar 3.5 Rancangan Tampilan Cropping

3.4.3. Rancangan Tampilan Akhir

Tampilan atau halaman akhir digunakan untuk menerapkan metode apakah sesuai template atau tidak, apabila sesuai maka akan membentuk *rectangle* berwarna hijau dan juga tulisan "Result : Matching" dan apabila tidak sesuai akan membentuk *rectangle* berwarna merah dengan tulisan "Result : No Matching".



Gambar 3.6 Rancangan Tampilan Akhir

3.5. Metode Pengujian *Black Box*

Rancangan pengujian, menggunakan metode pengujian *black box* (*black box testing*). *Black box* testing adalah salah satu metode pengujian perangkat lunak yang berfokus pada sisi fungsionalitas, khususnya pada *input* dan *output* aplikasi

(apakah sudah sesuai dengan apa yang diharapkan atau belum). Rencana pengujian sistem dan kelas yang di uji dapat dilihat pada tabel 3.1 di bawah ini.

Tabel 3.1 Rencana Pengujian

Kasus dan Hasil Uji			
Aktivitas Pengujian	**Realisasi yang diharapkan**	**Hasil Pengujian**	**Kesimpulan**
			[ ] Sesuai [ ] Tidak Sesuai

3.6. Cara Kerja Template Matching

Template matching membutuhkan dua komponen utama yaitu :

1. Gambar sumber (I) : Gambar yang diharapkan menemukan kecocokan dengan gambar template.
2. Gambar template (T) : Gambar template yang akan dibandingkan dengan gambar sumber.

   Tujuan nya adalah untuk mendeteksi area pencocokan tertinggi :

Gambar 3.7 Cara kerja 1

Untuk mengidentifikasi area yang cocok, harus membandingkan gambar template dengan gambar sumber dengan cara menggesernya :

Gambar 3.8 Cara Kerja 2

Menggeser, yang di maksud adalah memindahkan template satu piksel pada satu waktu (kiri ke kanan, atas ke bawah). Di setiap lokasi, metrik dihitung sehingga mewakili seberapa "baik" atau "buruk" kecocokan di lokasi tersebut (atau seberapa mirip patch dengan area tertentu dari gambar sumber).

Untuk setiap lokasi T di atas I, menyimpan metrik dalam matriks hasil R. Setiap lokasi (x,y) di R berisi metrik kecocokan :

Gambar 3.9 Cara Kerja 3

gambar di atas adalah hasil R dari sliding patch dengan metrik TM_CCORR_NORMED. Lokasi paling terang menunjukkan kecocokan tertinggi. Seperti yang dilihat, lokasi yang ditandai dengan lingkaran merah mungkin adalah yang memiliki nilai tertinggi, sehingga lokasi (persegi panjang yang dibentuk oleh

titik tersebut sebagai sudut dan lebar dan tinggi sama dengan gambar template) dianggap cocok.

Dalam praktiknya, kita menemukan nilai tertinggi (atau lebih rendah, tergantung pada jenis metode pencocokan) dalam matriks R, menggunakan fungsi minMaxLoc()

Cara kerja mask, dalam proses mask diperlukan 3 komponen :

1. Gambar sumber (I) : Gambar yang diharapkan menemukan kecocokan dengan gambar template.
2. Gambar template (T) : Gambar template yang akan dibandingkan dengan gambar sumber.
3. Gambar mask (M) : Mask, gambar skala abu-abu yang menutupi template.

Hanya dua metode pencocokan yang saat ini menerima mask : TM_SQDIFF dan TM_CCORR_NORMED (lihat di bawah untuk penjelasan tentang semua metode matching yang tersedia di opencv).

Mask harus memiliki dimensi yang sama dengan template.

Mask harus memiliki kedalaman CV_8U atau CV_32F dan jumlah saluran yang sama dengan gambar template. Dalam kasus CV_8U, nilai mask diperlakukan sebagai biner, yaitu nol dan bukan nol. Dalam kasus CV_32F, nilainya harus masuk ke dalam rentang [0..1] dan piksel templat akan dikalikan dengan nilai piksel topeng yang sesuai. Karena gambar masukan dalam sampel memiliki tipe CV_8UC3, topeng juga dibaca sebagai gambar berwarna.

OpenCV mengimplementasikan template matching dalam fungsi matchTemplate() dan ada 6 metode yang tersedia yaitu :

1. method=TM_SQDIFF

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

2. method=TM_SQDIFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

3. method=TM_CCORR

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

4. method=TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

5. method=TM_CCOEFF

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))$$

where

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

6. method=TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

Gambar 3.10 Metode Perhitungan

Fungsi dari program :

1. Memuat gambar input, tambalan gambar (template), dan mask secara opsional.
2. Pencocokan template Lakukan prosedur dengan menggunakan fungsi OpenCV matchTemplate() dengan salah satu dari 6 metode pencocokan yang dijelaskan sebelumnya. Pengguna dapat memilih metode dengan memasukkan

pilihannya di Trackbar. Jika masker disediakan, itu hanya akan digunakan untuk metode yang mendukung penyamaran.

3. Normalisasikan output dari prosedur pencocokan.
4. Lokalkan lokasi dengan probabilitas pencocokan yang lebih tinggi.
5. Membuat rectangle di sekitar area yang sesuai dengan kecocokan tertinggi.